

**THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re the Application of : Minoru TAKIMOTO

Filed : Concurrently herewith

For : INFORMATION PROCESSING SYSTEM  
ENABLING DYNAMICALLY LOADING OR  
REPLACING PROGRAM COMPONENT IN  
MEMORY ALLOCATED TO ACTIVATED  
PROCESS

Serial No. : Concurrently herewith

December 4, 2000

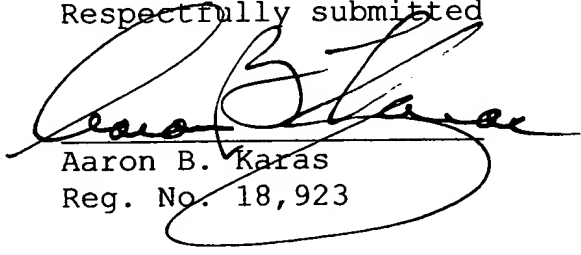
Assistant Commissioner of Patents  
Washington, D.C. 20231

**SUBMISSION OF PRIORITY DOCUMENT**

S I R:

Attached herewith is Japanese patent application No.  
2000-077814 of March 15, 2000 whose priority has been claimed in  
the present application.

Respectfully submitted

  
Aaron B. Karas  
Reg. No. 18,923

HELFGOTT & KARAS, P.C.  
60th FLOOR  
EMPIRE STATE BUILDING  
NEW YORK, NY 10118  
DOCKET NO.: FUJR18.034  
LHH:priority

Filed Via Express Mail  
Rec. No.: EL522396423US  
On: December 4, 2000  
By: Lydia Gonzalez

Any fee due with this paper, not fully  
covered by an enclosed check, may be  
charged on Deposit Acct. No. 08-1634

JCS60 U.S. PTO  
09/729015



#3  
9-22-01

日 本 国 特 許 庁  
PATENT OFFICE  
JAPANESE GOVERNMENT

JCS960 U.S. PTO  
09/729015  
12/04/06

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

2000年 3月15日

出 願 番 号

Application Number:

特願2000-077814

出 願 人

Applicant (s):

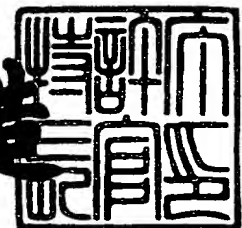
富士通株式会社

CERTIFIED COPY OF  
PRIORITY DOCUMENT

2000年 8月25日

特許庁長官  
Commissioner,  
Patent Office

及 川 耕 造



出証番号 出証特2000-3068205

【書類名】 特許願

【整理番号】 9952080

【提出日】 平成12年 3月15日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 09/06

【発明の名称】 プログラム置換システム、分散処理システム及びプログラム置換方法

【請求項の数】 28

【発明者】

    【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

    【氏名】 瀧本 稔

【特許出願人】

    【識別番号】 000005223

    【氏名又は名称】 富士通株式会社

【代理人】

    【識別番号】 100092152

    【弁理士】

    【氏名又は名称】 服部 毅巖

    【電話番号】 0426-45-6644

【手数料の表示】

    【予納台帳番号】 009874

    【納付金額】 21,000円

【提出物件の目録】

    【物件名】 明細書 1

    【物件名】 図面 1

    【物件名】 要約書 1

    【包括委任状番号】 9705176

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 プログラム置換システム、分散処理システム及びプログラム置換方法

【特許請求の範囲】

【請求項 1】 プログラムの置換を行うプログラム置換システムにおいて、プログラムの版数情報を含む電文を送信する電文送信手段を有する端末装置と

置換用のプログラムを格納するプログラム格納手段と、前記プログラムの情報を管理する転送前情報管理テーブルと、前記プログラム格納手段から転送されたプログラムを一時格納するプログラム一時格納手段と、前記プログラム一時格納手段に格納されている前記プログラムの情報を管理する転送後情報管理テーブルと、前記電文に含まれる前記版数情報に対応するプログラムを、起動中のプログラムが存在するプロセス内の前記プログラム一時格納手段に動的にリンクし、前記プログラムを実行するプログラム実行手段と、から構成されるプログラム置換装置と、

を有することを特徴とするプログラム置換システム。

【請求項 2】 前記転送後情報管理テーブルは、前記プログラム一時格納手段に格納されているプログラムの参照回数をカウントする参照カウンタをテーブル項目として有することを特徴とする請求項 1 記載のプログラム置換システム。

【請求項 3】 前記プログラム実行手段は、プログラムに対する前記参照カウンタの値が 0 で、前記プログラムの版数情報より新しい版数情報が前記転送後情報管理テーブルにある場合には、前記プログラムを前記プログラム一時格納手段から解放することを特徴とする請求項 2 記載のプログラム置換システム。

【請求項 4】 前記電文送信手段は、プログラムの前記版数情報と、前記プログラムが評価中か否かを示す評価フラグと、を含む電文を送信することを特徴とする請求項 1 記載のプログラム置換システム。

【請求項 5】 前記プログラム実行手段は、前記評価フラグが評価中を示す場合には、別プロセスの中で評価対象のプログラムを実行することを特徴とする請求項 4 記載のプログラム置換システム。

【請求項6】 前記転送後情報管理テーブルは、前記プログラム一時格納手段に格納されているプログラムの評価回数をカウントする評価カウンタをテーブル項目として有することを特徴とする請求項4記載のプログラム置換システム。

【請求項7】 前記プログラム実行手段は、プログラムに対する前記評価カウンタの値が0の場合は、前記別プロセスの中の前記プログラムの実行を中止することを特徴とする請求項6記載のプログラム置換システム。

【請求項8】 前記プログラム及び前記プログラムに関する情報の設定・管理を一括して行う管理装置を有することを特徴とする請求項1記載のプログラム置換システム。

【請求項9】 ソフトウェアの処理を行う端末装置において、  
前記ソフトウェアの入出力制御を行うインタフェース手段と、  
プログラムの版数情報を含む電文を送信する電文送信手段と、  
を有することを特徴とする端末装置。

【請求項10】 プログラムの置換を行うプログラム置換装置において、  
置換用のプログラムを格納する置換用プログラム格納手段と、  
前記プログラム格納手段に格納されている前記プログラムの情報を管理する転送前情報管理テーブルと、

前記プログラム格納手段から転送されたプログラムを一時格納するプログラム一時格納手段と、

前記プログラム一時格納手段に格納されている前記プログラムの情報を管理する転送後情報管理テーブルと、

前記電文に含まれる前記版数情報に対応するプログラムを、起動中のプログラムが存在するプロセス内の前記プログラム一時格納手段に動的にリンクし、前記プログラムを実行するプログラム実行手段と、

を有することを特徴とするプログラム置換装置。

【請求項11】 n階層のクライアント／サーバ環境で分散処理を行う分散処理システムにおいて、

分散オブジェクトの版数情報を含む電文のスタブ処理を行うクライアント側スタブ処理手段を有するクライアント装置と、

前記電文のスケルトン処理を行うサーバ側スケルトン処理手段と、置換用の分散オブジェクトを格納する分散オブジェクト格納リポジトリと、前記分散オブジェクトの情報を管理する転送前情報管理テーブルと、前記分散オブジェクト格納リポジトリから転送された分散オブジェクトを一時格納するメモリと、前記メモリに格納されている前記分散オブジェクトの情報を管理する転送後情報管理テーブルと、前記電文に含まれる前記版数情報に対応する分散オブジェクトを、起動中の分散オブジェクトが存在するプロセス内の前記メモリに動的にリンクし、前記分散オブジェクトの関数を呼び出して実行する分散オブジェクト呼び出し制御手段と、他装置内にある分散オブジェクトの関数を呼び出して実行するために、前記版数情報を含む電文のスタブ処理を行うサーバ側スタブ処理手段と、から構成され、装置間で同期して前記分散オブジェクトの置換を行う複数のサーバ装置と、

を有することを特徴とする分散処理システム。

【請求項 1 2】 前記転送後情報管理テーブルは、前記メモリに格納されている分散オブジェクトの参照回数をカウントする参照カウンタをテーブル項目として有することを特徴とする請求項 1 1 記載の分散処理システム。

【請求項 1 3】 前記分散オブジェクト呼び出し制御手段は、分散オブジェクトに対する前記参照カウンタの値が 0 で、前記分散オブジェクトの版数情報より新しい版数情報が前記転送後情報管理テーブルにある場合には、前記分散オブジェクトを前記メモリから解放することを特徴とする請求項 1 2 記載の分散処理システム。

【請求項 1 4】 前記電文送信手段は、分散オブジェクトの前記版数情報と、前記分散オブジェクトが評価中か否かを示す評価フラグと、を含む電文を送信することを特徴とする請求項 1 1 記載の分散処理システム。

【請求項 1 5】 前記分散オブジェクト呼び出し制御手段は、前記評価フラグが評価中を示す場合には、別プロセスの中で評価対象の分散オブジェクトの関数を呼び出して実行することを特徴とする請求項 1 4 記載の分散処理システム。

【請求項 1 6】 前記転送後情報管理テーブルは、前記メモリに格納されている分散オブジェクトの評価回数をカウントする評価カウンタをテーブル項目と

して有することを特徴とする請求項 1 4 記載の分散処理システム。

【請求項 1 7】 前記分散オブジェクト呼び出し制御手段は、分散オブジェクトに対する前記評価カウンタの値が 0 の場合は、前記別プロセスの中の前記分散オブジェクトの実行を中止することを特徴とする請求項 1 6 記載の分散処理システム。

【請求項 1 8】 前記分散オブジェクト及び前記分散オブジェクトに関する情報の設定・管理を一括して行う管理サーバを有することを特徴とする請求項 1 1 記載の分散処理システム。

【請求項 1 9】 サービスの依頼を行うクライアント装置において、  
ソフトウェアの入出力制御を行うインタフェース手段と、  
分散オブジェクトの版数情報を含む電文のスタブ処理を行うクライアント側スタブ処理手段と、  
を有することを特徴とするクライアント装置。

【請求項 2 0】 サービスの提供を行うサーバ装置において、  
送信された電文のスケルトン処理を行うサーバ側スケルトン処理手段と、  
置換用の分散オブジェクトを格納する分散オブジェクト格納リポジトリと、  
前記分散オブジェクトの情報を管理する転送前情報管理テーブルと、  
前記分散オブジェクト格納リポジトリから転送された分散オブジェクトを一時格納するメモリと、

前記メモリに格納されている前記分散オブジェクトの情報を管理する転送後情報管理テーブルと、

前記電文に含まれる前記版数情報に対応する分散オブジェクトを、起動中の分散オブジェクトが存在するプロセス内の前記メモリに動的にリンクし、前記分散オブジェクトの関数を呼び出して実行する分散オブジェクト呼び出し制御手段と、

他装置内にある分散オブジェクトを呼び出して実行するために、前記版数情報を含む電文のスタブ処理を行うサーバ側スタブ処理手段と、

を有することを特徴とするサーバ装置。

【請求項 2 1】 n 階層のクライアント／サーバ環境でプログラムの置換を

行うプログラム置換方法において、

クライアントからプログラムの版数情報を含む電文を送信し、

前記電文をサーバで受信し、

置換用のプログラムを格納し、

前記プログラムの情報を転送後情報管理テーブルで管理し、

前記電文に含まれる前記版数情報に対応するプログラムを、起動中のプログラムが存在するプロセス内のメモリに動的にリンクして、前記プログラムを呼び出して実行し、

前記メモリに格納されている前記プログラムの情報を転送後情報管理テーブルで管理し、

他装置内にあるプログラムを呼び出して実行するために、前記版数情報を含む電文の配信処理を行うことにより、n階層間でのプログラムの置換を、装置間で同期させて行うプログラム置換方法。

【請求項22】 前記転送後情報管理テーブルは、前記メモリに格納されているプログラムの参照回数をカウントする参照カウンタをテーブル項目として有することを特徴とする請求項21記載のプログラム置換方法。

【請求項23】 プログラムに対する前記参照カウンタの値が0で、前記プログラムの版数情報より新しい版数情報が前記転送後情報管理テーブルにある場合には、前記プログラムを前記メモリから解放することを特徴とする請求項22記載のプログラム置換方法。

【請求項24】 プログラムの前記版数情報と、前記プログラムが評価中か否かを示す評価フラグと、を含む電文を送信することを特徴とする請求項21記載のプログラム置換方法。

【請求項25】 前記評価フラグが評価中を示す場合には、別プロセスの中で評価対象のプログラムを呼び出して実行することを特徴とする請求項24記載のプログラム置換方法。

【請求項26】 前記転送後情報管理テーブルは、前記メモリに格納されているプログラムの評価回数をカウントする評価カウンタをテーブル項目として有することを特徴とする請求項24記載のプログラム置換方法。



【請求項 2 7】 プログラムに対する前記評価カウンタの値が 0 の場合は、前記別プロセスの中の前記プログラムの実行を中止することを特徴とする請求項 2 6 記載のプログラム置換方法。

【請求項 2 8】 管理サーバにより、置換用の前記プログラム及び前記プログラムに関する情報の設定・管理を一括して行うことを特徴とする請求項 2 1 記載のプログラム置換方法。

【発明の詳細な説明】

【0 0 0 1】

【発明が属する技術分野】

本発明はプログラム置換システム、分散処理システム及びプログラム置換方法に関し、特にプログラムの置換を行うプログラム置換システム、 $n$ 階層のクライアント／サーバ環境で分散処理を行う分散処理システム及び $n$ 階層のクライアント／サーバ環境でプログラムの置換を行うプログラム置換方法に関する。

【0 0 0 2】

【従来の技術】

近年、通信ネットワークは、多種多様な機能が求められている。このような通信ネットワークに対し、多様なサービスを顧客に提供したり、ネットワークの保守・運用を制御するためには、ソフトウェアのプログラムの版数を上げて、常に改版したプログラムの入れ替えを行う必要がある。

【0 0 0 3】

また、特に基幹のネットワークでは、プロセスまたはシステム全体を一旦停止するといったことはできないため、システムの稼働中に動的にプログラムの置換を行わねばならない。

【0 0 0 4】

従来のプログラムの動的置換技術としては、例えば、新版プログラムをキューイングさせて、旧版プログラムと置換した後に通信を再開させたり、あるいは、旧版プログラムに対する要求を一旦キャンセルして、新版プログラムとの置換を行ったりしている。

【0 0 0 5】

## 【発明が解決しようとする課題】

しかし、上記のような従来のプログラムの動的置換技術では、プロセスまたはシステム全体を一旦停止する必要はないものの、一時的な業務処理の閉塞や中断が発生してしまうといった問題があった。

## 【0006】

一方、ソフトウェアの処理形態として、クライアント／サーバに代表される分散処理形態が、通信ネットワークに対しても広く活用されており、近年では、クライアント層とサーバ層の間に、他の独立した層を複数設けたn階層のクライアント／サーバが、次世代のクライアント／サーバ・アーキテクチャを特徴づけるものとして開発が進んでいる。

## 【0007】

従来のような新、旧2つのプログラムの入れ替えといった単純な場合のみ想定しているプログラムの動的置換技術では、このようなn階層クライアント／サーバ環境に対し、複数のコンピュータ間で複数のソフトウェアを同期させながら動的に入れ替えていくことは困難である。

## 【0008】

本発明はこのような点に鑑みてなされたものであり、業務の中断なしにソフトウェアを動的に効率よく置換するプログラム置換システムを提供することを目的とする。

## 【0009】

また、本発明の他の目的は、クライアント／サーバ環境で、業務の中断なしにソフトウェアを動的に効率よく置換する分散処理システムを提供することである。

## 【0010】

さらに、本発明の他の目的は、クライアント／サーバ環境で、業務の中断なしにソフトウェアを動的に効率よく置換、配信するプログラム置換方法を提供することである。

## 【0011】

## 【課題を解決するための手段】

本発明では上記課題を解決するために、図 1 のような、プログラムの置換を行うプログラム置換システム 1 において、プログラムの版数情報を含む電文を送信する電文送信手段 1 1 を有する端末装置 1 0 と、置換用のプログラムを格納するプログラム格納手段 2 1 と、プログラムの情報を管理する転送前情報管理テーブル T 2 1 と、プログラム格納手段 2 1 から転送されたプログラムを一時格納するプログラム一時格納手段 2 2 と、プログラム一時格納手段 2 2 に格納されているプログラムの情報を管理する転送後情報管理テーブル T 2 2 と、電文に含まれる版数情報に対応するプログラムを、起動中のプログラムが存在するプロセス P a 内のプログラム一時格納手段 2 2 に動的にリンクし、プログラムを実行するプログラム実行手段 2 3 と、から構成されるプログラム置換装置 2 0 と、を有することを特徴とするプログラム置換システム 1 が提供される。

#### 【 0 0 1 2 】

ここで、電文送信手段 1 1 は、プログラムの版数情報を含む電文を送信する。プログラム格納手段 2 1 は、置換用のプログラムを格納する。転送前情報管理テーブル T 2 1 は、プログラムの情報を管理する。プログラム一時格納手段 2 2 は、プログラム格納手段 2 1 から転送されたプログラムを一時格納する。転送後情報管理テーブル T 2 2 は、プログラム一時格納手段 2 2 に格納されているプログラムの情報を管理する。プログラム実行手段 2 3 は、電文に含まれる版数情報に対応するプログラムを、起動中のプログラムが存在するプロセス P a 内のプログラム一時格納手段 2 2 に動的にリンクし、プログラムを実行する。

#### 【 0 0 1 3 】

また、図 3 に示すような n 階層のクライアント／サーバ環境で分散処理を行う分散処理システム 1 a において、分散オブジェクトの版数情報を含む電文のスタブ処理を行うクライアント側スタブ処理手段 3 1 を有するクライアント装置 3 0 と、電文のスケルトン処理を行うサーバ側スケルトン処理手段 4 4 と、置換用の分散オブジェクトを格納する分散オブジェクト格納リポジトリ 4 1 と、分散オブジェクトの情報を管理する転送前情報管理テーブル T 1 と、分散オブジェクト格納リポジトリ 4 1 から転送された分散オブジェクトを一時格納するメモリ 4 2 と、メモリ 4 2 に格納されている分散オブジェクトの情報を管理する転送後情報管

理テーブルT2と、電文に含まれる版数情報に対応する分散オブジェクトを、起動中の分散オブジェクトが存在するプロセスP1内のメモリ42に動的にリンクし、分散オブジェクトの関数を呼び出して実行する分散オブジェクト呼び出し制御手段43と、他装置内にある分散オブジェクトを呼び出して実行するために、版数情報を含む電文のスタブ処理を行うサーバ側スタブ処理手段45と、から構成され、装置間で同期して分散オブジェクトの置換を行う複数のサーバ装置40-1~40-nと、を有することを特徴とする分散処理システム1aが提供される。

#### 【0014】

ここで、クライアント側スタブ処理手段31は、分散オブジェクトの版数情報を含む電文のスタブ処理を行う。サーバ側スケルトン処理手段44は、電文のスケルトン処理を行う。分散オブジェクト格納リポジトリ41は、置換用の分散オブジェクトを格納する。転送前情報管理テーブルT1は、分散オブジェクトの情報を管理する。メモリ42は、分散オブジェクト格納リポジトリ41から転送された分散オブジェクトを一時格納する。転送後情報管理テーブルT2は、メモリ42に格納されている分散オブジェクトの情報を管理する。分散オブジェクト呼び出し制御手段43は、電文に含まれる版数情報に対応する分散オブジェクトを、起動中の分散オブジェクトが存在するプロセスP1内のメモリ42に動的にリンクし、分散オブジェクトの関数を呼び出して実行する。サーバ側スタブ処理手段45は、他装置内にある分散オブジェクトを呼び出して実行するために、版数情報を含む電文のスタブ処理を行う。

#### 【0015】

さらに、図20のような、n階層のクライアント／サーバ環境でプログラムの置換を行うプログラム置換方法において、クライアントからプログラムの版数情報を含む電文を送信し、電文をサーバで受信し、置換用のプログラムを格納し、プログラムの情報を転送前情報管理テーブルで管理し、電文に含まれる版数情報に対応するプログラムを、起動中のプログラムが存在するプロセス内のメモリに動的にリンクして、プログラムを呼び出して実行し、メモリに格納されているプログラムの情報を転送後情報管理テーブルで管理し、他装置内にあるプログラム

を呼び出して実行するために、版数情報を含む電文の配信処理を行うことにより、 $n$ 階層間でのプログラムの置換を、装置間で同期させて行うプログラム置換方法が提供される。

【0016】

ここで、プログラムの版数情報を含む電文を送信し、この版数情報に対応するプログラムをプロセス内のメモリに動的にリンクして実行する。

【0017】

【発明の実施の形態】

以下、本発明の実施の形態を図面を参照して説明する。図1は本発明のプログラム置換システムの原理図である。プログラム置換システム1は、端末装置10とプログラム置換装置20から構成され、ネットワーク100等を介して接続する。

【0018】

端末装置10に対し、電文送信手段11は、置換したいアプリケーション・プログラム（以下、プログラム）の版数情報を含む業務電文（以下、電文）を送信する。

【0019】

図では、版数情報（1.03）を含む電文を、プログラム置換装置20へ送信している。版数情報の値が大きいものほど新しい版数である。なお、端末装置10は、図に示さないインタフェース手段を有しており、インタフェース手段はソフトウェアの入出力制御を行う。

【0020】

プログラム格納手段21は、置換用のプログラムを格納する。転送前情報管理テーブルT21は、プログラム格納手段21に格納されているプログラムの情報を管理する。管理情報の項目としては例えば、版数情報や、その版数情報に対応したプログラム名等がある。

【0021】

プログラム一時格納手段22は、プロセスPa内のメモリであり、プログラム格納手段21から転送されたプログラムを一時格納する。図では、版数情報が（

1. 0 1) ~ (1. 0 3) のプログラムが転送されて格納されている。転送後情報管理テーブル T 2 2 は、プログラム一時格納手段 2 2 に格納されているプログラムの情報を管理する。

【0 0 2 2】

プログラム実行手段 2 3 は、電文に含まれる版数情報に対応するプログラムを、起動中のプログラムが存在するプロセス P a 内のプログラム一時格納手段 2 2 に動的にリンクし、プログラムを実行する。

【0 0 2 3】

なお、動的にリンクするとは、すでに起動しているプログラムのトランザクション処理等を停止せずに、そのプログラムが存在するプロセスと同一のプロセス内にあるメモリへプログラムを転送することをいう。

【0 0 2 4】

ここで、プログラム格納手段 2 1 からプログラム一時格納手段 2 2 へ、プログラム (1. 0 1)、(1. 0 2) がすでに転送されているものとする。

その後、(1. 0 3) の版数情報を含む電文を受信すると、プログラム実行手段 2 3 は、転送前情報管理テーブル T 2 1、転送後情報管理テーブル T 2 2 へアクセスして、プログラムの存在場所を確認し (図 2 で後述)、リンクしていない場合には、プログラム一時格納手段 2 2 へプログラム (1. 0 3) を転送する。

【0 0 2 5】

これにより、同一プロセス P a 内にプログラム (1. 0 3) が組み込まれ、新しいプログラムの処理を開始することができる。

このように、本発明では、すでに起動しているプログラムに対して、動的に新しいプログラムを組み込むことができるので、業務を中断せずに、プログラムの置換を行うことが可能になる。

【0 0 2 6】

なお、プログラム (1. 0 3) が起動中に、旧版のプログラム (1. 0 1) を起動させたい場合には、電文に版数情報 (1. 0 1) を含めて送信する。そして、プログラム実行手段 2 3 が、プログラム一時格納手段 2 2 にすでに転送済みのプログラム (1. 0 1) を呼び出して実行すればよい。

## 【 0 0 2 7 】

一方、ネットワーク 1 0 0 には、ネットワーク管理装置 5 0 が接続する。ネットワーク管理装置 5 0 は、ネットワーク 1 0 0 に接続する装置の保守・運用を行う。本発明に関していえば、例えば、プログラム格納手段 2 1 に格納するプログラムや転送前情報管理テーブル T 2 1 に記載する各種情報の設定・管理を一括して行う。

## 【 0 0 2 8 】

次にフローチャートを用いて動作について説明する。図 2 はプログラム置換システム 1 の動作手順を示すフローチャートである。

〔 S 1 〕プログラム置換装置 2 0 は、端末装置 1 0 から送信された版数情報（版数情報 A とする）を含む電文を受信する。

〔 S 2 〕プログラム実行手段 2 3 は、転送後情報管理テーブル T 2 2 に対し、版数情報 A があるか否かを判断する。あればステップ S 3 へ、なければステップ S 4 へ行く。

〔 S 3 〕プログラム実行手段 2 3 は、版数情報 A のプログラムを実行する。

〔 S 4 〕プログラム実行手段 2 3 は、転送前情報管理テーブル T 2 1 を参照して、版数情報 A があるか否かを判断する。あればステップ S 6 へ、なければステップ S 5 へ行く。

〔 S 5 〕プログラム置換装置 2 0 は、端末装置 1 0 へエラーを送信する。

〔 S 6 〕プログラム実行手段 2 3 は、プログラム格納手段 2 1 からプログラム一時格納手段 2 2 へ、版数情報 A のプログラムを動的にリンクする。

〔 S 7 〕プログラム実行手段 2 3 は、転送後情報管理テーブル T 2 2 の内容を変更する。

〔 S 8 〕プログラム実行手段 2 3 は、版数情報 A のプログラムを実行する。

## 【 0 0 2 9 】

次に本発明のプログラム置換システム 1 を n 階層のクライアント／サーバ環境のネットワークに適用した際の分散処理システムについて説明する。

図 3 は分散処理システムの構成を示す図である。第 1 の実施の形態である分散処理システム 1 a は、クライアント装置 3 0 と複数のサーバ装置 4 0 - 1 ~ 4 0

-nとから構成される。クライアント装置30と複数のサーバ装置40-1~40-nは、分散オブジェクト・コンピューティングを基本とするn階層クライアント/サーバ・アーキテクチャを持つネットワーク200上に配置される。

【0030】

また、ネットワーク200上には、CORBA (Common Object Request Broker Architecture)やDCOM (Distributed Component Object Model)等の分散オブジェクト通信インフラストラクチャが準備されるが(以降、分散オブジェクト通信インフラストラクチャをORB (Object Request Broker)と呼ぶ)、本発明はCORBAのORBが適用されたシステムとして以降説明する。

【0031】

まず、ネットワーク200上に接続するWebサーバ60からクライアント・アプリケーションが、クライアント装置30へダウンロードされる。その後、クライアント装置30と複数のサーバ装置40-1~40-nに対して、CORBAのORB上で分散オブジェクトの配信、置換が行われる。ここで、分散オブジェクトとは、個々の機能単位を指しており、アプリケーション・プログラム、ライブラリ等に該当する。

【0032】

クライアント装置30に対し、クライアント側スタブ処理手段31は、分散オブジェクトの版数情報を含む電文のスタブ処理を行う。スタブ処理とは、ORB上でクライアントからサーバへ分散オブジェクトを送信する際の、クライアント側での送信処理のことをいう。

【0033】

なお、クライアント装置30は、ソフトウェアの入出力制御を行うインタフェース手段(図示せず)を有する。

サーバ装置40-1に対し(サーバ装置40-2~40-nもサーバ装置40-1と同機能を持つ)、サーバ側スケルトン処理手段44は、電文のスケルトン処理を行う。スケルトン処理とは、ORB上でクライアントから送信された分散オブジェクトをサーバで受信する際の、サーバ側での受信処理のことをいう。

【0034】



分散オブジェクト格納リポジトリ 4 1 は、置換用の分散オブジェクトを格納する。転送前情報管理テーブル T 1 は、分散オブジェクトの情報を管理する。管理情報の詳細な項目は図 8 で後述する。

#### 【 0 0 3 5 】

メモリ 4 2 は、分散オブジェクト格納リポジトリ 4 1 から転送された分散オブジェクトを一時格納するプロセス内メモリである。転送後情報管理テーブル T 2 は、メモリ 4 2 に格納されている分散オブジェクトの情報を管理する。管理情報の詳細な項目は図 1 0 で後述する。

#### 【 0 0 3 6 】

分散オブジェクト呼び出し制御手段 4 3 は、電文に含まれる版数情報に対応する分散オブジェクトを、起動中の分散オブジェクトが存在するプロセス P 1 内のメモリ 4 2 に動的にリンクする。そして、分散オブジェクトの関数を呼び出して実行する。

#### 【 0 0 3 7 】

サーバ側スタブ処理手段 4 5 は、他装置（サーバ装置 4 0 - 2 ~ 4 0 - n）内にある分散オブジェクトを呼び出して実行するために、版数情報を含む電文のスタブ処理を行う（この場合、サーバ装置 4 0 - 1 はクライアントとなる）。

#### 【 0 0 3 8 】

一方、ネットワーク 2 0 0 には、管理サーバ 5 0 a が接続する。管理サーバ 5 0 a は、ネットワーク 2 0 0 に接続する装置の保守・運用を行う。例えば、分散オブジェクト格納リポジトリ 4 1 に格納する分散オブジェクトや転送前情報管理テーブル T 1 に記載する各種情報の設定・管理を一括して行う。

#### 【 0 0 3 9 】

ここで、分散オブジェクト格納リポジトリ 4 1 からメモリ 4 2 へ、分散オブジェクト（1. 0 1）、（1. 0 2）がすでに転送されているものとする。

その後、（1. 0 3）の版数情報を含む電文を受信すると、分散オブジェクト呼び出し制御手段 4 3 は、転送前情報管理テーブル T 1、転送後情報管理テーブル T 2 へアクセスして、分散オブジェクトの存在場所を確認し（図 1 1 で後述）、リンクしていない場合には、メモリ 4 2 へ分散オブジェクト（1. 0 3）を転

送する。

【0040】

これにより、同一プロセスP1内に分散オブジェクト(1.03)が組み込まれ、新しい分散オブジェクトの処理を開始することができる。

また、この場合、サーバ側スタブ処理手段45は、版数情報(1.03)を含む電文を生成して、サーバ装置40-2へ配信し、サーバ装置40-2でも分散オブジェクト(1.03)が起動し、その後、サーバ装置40-2は版数情報(1.03)を含む電文をサーバ装置40-3へ配信する。以降同様にして、ネットワーク200上のすべてのサーバ装置に対して、分散オブジェクト(1.03)が設定されて起動することになる。

【0041】

このように、本発明では、旧分散オブジェクトのトランザクション処理中に、ネットワーク200上のサーバ装置40-1～40-n間で同期させながら、新分散オブジェクトのトランザクション処理に入れ替えていく構成とした。これにより、業務を中断せずに、ネットワーク200上のすべての装置に対して、分散オブジェクトの配信、置換を自動的に効率よく行うことが可能になり、システムの信頼性及び品質の向上を図ることが可能になる。

【0042】

次にサーバ装置40(サーバ装置40-1～40-nを総称してサーバ装置40とする)の詳細な構成について説明する。

図4はサーバ装置40の構成を示す図である。スケルトン組み込み機構44aとスケルトン処理手段44bは、サーバ側スケルトン処理手段44に対応し、スタブ組み込み機構45aとスタブ処理手段45bは、サーバ側スタブ処理手段45に対応する。また、コンポーネント版数情報定義ファイルT1は転送前情報管理テーブルT1に対応し、コンポーネント版数情報管理テーブルT2は転送後情報管理テーブルT2に対応する。

【0043】

さらに、以降の説明では、分散オブジェクトを主体性を持つ存在として見た場合に、その状態及び状態の変化を分散オブジェクト振る舞い処理と呼ぶ。

スケルトン格納リポジトリ 4 6 は、サーバ側スケルトン処理用共有ライブラリを格納する。スケルトン組み込み機構 4 4 a は、コンポーネント版数情報定義ファイル T 1 にアクセスして、コンポーネント版数情報定義ファイル T 1 で管理されているプロセス名を認識する。

【 0 0 4 4 】

そして、スケルトン組み込み機構 4 4 a は、このプロセス名に対応したサーバ側スケルトン処理用共有ライブラリ L a を、スケルトン格納リポジトリ 4 6 から呼び出す。その後、サーバ側スケルトン処理用共有ライブラリ L a のスケルトン処理手段 4 4 b で電文の受信処理を行う。

【 0 0 4 5 】

分散オブジェクト格納リポジトリ 4 1 は、分散オブジェクトの振る舞い処理用共有ライブラリを格納する。分散オブジェクト呼び出し制御手段 4 3 は、電文に含まれる版数情報に対応する分散オブジェクトを、コンポーネント版数情報定義ファイル T 1、コンポーネント版数情報管理テーブル T 2 へアクセスして、分散オブジェクトの存在場所を確認し（図 1 1 で後述）、リンクしていない場合には、メモリ 4 2 へ分散オブジェクト振る舞い処理（の共有ファイル）を転送（以下、ロード）し、分散オブジェクトの関数を呼び出して実行する。また、動的リンクの状況や分散オブジェクトの各関数へのポインタを管理する。

【 0 0 4 6 】

スタブ格納リポジトリ 4 7 は、クライアント側スタブ処理用共有ライブラリを格納する。スタブ組み込み機構 4 5 a は、他装置の分散オブジェクトをさらに呼び出す場合に、スタブ処理用共有ライブラリ L b をスタブ格納リポジトリ 4 7 から呼び出す。そして、スタブ処理用共有ライブラリ L b のスタブ処理手段 4 5 b で電文の送信処理を行う。

【 0 0 4 7 】

スタブ処理の詳細な動作としては、まず、分散オブジェクト振る舞い処理中に、さらに他サーバ装置のサーバ・アプリケーションへ、処理を要求する必要がある場合は、スタブ組み込み機構 4 5 a を実行する。

【 0 0 4 8 】

そして、業務依頼をするサーバ・アプリケーションとの通信を確立するため、スタブ格納リポジトリ 4 7 から該当するスタブ処理用共有ライブラリ L b を取得し、動的にリンクする。

【 0 0 4 9 】

スタブ処理手段 4 5 b では、サーバ・アプリケーションとの接続処理を実施する。また、分散オブジェクト振る舞い処理では、サーバとの接続を待ち、システム全体の版数情報を業務電文に含めて送信する。このような動作を行うことにより、サーバ装置 4 0 - 1 ~ 4 0 - n 間で、分散オブジェクトの配信、置換を同期させながら行うことが可能になる。

【 0 0 5 0 】

なお、プロセスは、プロセス P 1 ~ P n のように複数生成することができ、プロセス P 1 ~ P n 内の処理と、スケルトン格納リポジトリ 4 6 とスタブ格納リポジトリ 4 7 と分散オブジェクト格納リポジトリ 4 1 とコンポーネント版数情報定義ファイル T 1 とは、オペレーティングシステム 4 8 を介して通信を行う。

【 0 0 5 1 】

さらに、スケルトン格納リポジトリ 4 6、スタブ格納リポジトリ 4 7、分散オブジェクト格納リポジトリ 4 1 に対する、共有ライブラリの配信は、図 3 で上述した管理サーバ 5 0 a により行われ、集中管理が可能である。

【 0 0 5 2 】

次に電文の構成について説明する。図 5 は電文の構成例を示す図である。電文 2 は、分散オブジェクトの関数名 2 a とパラメタ 2 c と、本発明で付加したシステム全体の版数を示す版数情報（メジャー番号+マイナー番号） 2 b から構成される。

【 0 0 5 3 】

図 6 は電文の I D L 定義例を示す図である。I D L (Interface Definition Language) は、分散オブジェクトのインタフェースを定義する言語である。すべての interface について、他パラメタに先立ち、版数情報を in パラメタとして定義する。

【 0 0 5 4 】

図7はIDL定義からC++言語へのマッピング例を示す図である。(A)は版数情報(1.01)であり、(B)は版数情報(1.02)である。

(A)、(B)に示すように、本発明では、分散オブジェクトのモジュール名に版数情報を付加したネームスペースNa、Nbを付加している。従来では、IDLと分散オブジェクト振る舞い処理とは1:1対応であったが、本発明ではC++言語のネームスペースを使用することにより、複数の版数の分散オブジェクトの関数名、変数名の衝突を防ぐことができ(すなわち、(A)のclass XXX、と(B)のclass XXXを全く別のものとして定義できる)、複数の版数の分散オブジェクト振る舞い処理用の共有ライブラリを、同一プロセスのメモリに動的にリンクすることができる。

#### 【0055】

次にコンポーネント版数情報定義ファイルT1について説明する。図8はコンポーネント版数情報定義ファイルT1の構成を示す図である。コンポーネント版数情報定義ファイルT1は、システム全体の版数情報T1a、分散オブジェクト振る舞い処理用共有ライブラリ名T1b、共有ライブラリの版数情報T1c、プロセス名T1dから構成される。

#### 【0056】

システム全体の版数情報T1aは、クライアント装置30から要求される版数情報に対応する。分散オブジェクト振る舞い処理用共有ライブラリ名T1bは、版数情報に対応するライブラリ名である。例えば図では、版数情報(1.01)に対して、libAPL-A.so ~ libAPL-X.soが対応している。

#### 【0057】

また、同じライブラリ名で異なる内容の共有ライブラリが、スケルトン格納リポジトリ46やスタブ格納リポジトリ47に格納されている。例えば、スケルトン格納リポジトリ46に格納されているlibAPL-A.soは、プロセス(process-1)で用いるスケルトン処理用のライブラリ(スケルトン処理用共有ライブラリLa)であり、スタブ格納リポジトリ47に格納されているlibAPL-A.soは、プロセス(process-1)で用いるスタブ処理用のライブラリ(スタブ処理用共有ライブラリLb)である。

## 【0058】

共有ライブラリの版数情報T1cは、分散オブジェクトのコンポーネントの版数情報である。共有ライブラリの版数情報T1cでは、複数の版数を持つ複数のソフトウェア・コンポーネントの組み合わせをシステム全体の版数で関連付けているため、システム全体の版数情報から各分散オブジェクトの振る舞い処理コンポーネントの版数が一意に決定することができる。

## 【0059】

プロセス名T1dは、リンク先のプロセスのことであり、どのプロセスで実行するかが示される。

次にリポジトリ等のディレクトリ構成について説明する。図9はディレクトリの構成例を示す図である。／opt、／aptの下位に／etc、／bin、／skel、／stub、／objがある。

## 【0060】

／etcはコンポーネント版数情報定義ファイルT1を格納し、／binはスケルトン組み込み機構44aを格納し、／skelはスケルトン格納リポジトリ46の格納領域であり、／stubはスタブ格納リポジトリ47の格納領域である。

## 【0061】

また、／objの下位に／1.01、／1.02、…があり、これらは版数情報毎に対応した分散オブジェクト格納リポジトリ41の格納領域である。

次にコンポーネント版数情報管理テーブルT2について説明する。図10はコンポーネント版数情報管理テーブルT2の構成を示す図である。

## 【0062】

コンポーネント版数情報管理テーブルT2は、メモリ42にロードされた分散オブジェクト振る舞い処理の情報を格納する。項目としては、システム全体の版数情報T2a、分散オブジェクト振る舞い処理用共有ライブラリ名T2b、共有ライブラリの版数情報T2c、フラグT2d、関数テーブルへのアドレスT2eから構成される。システム全体の版数情報T2a、分散オブジェクト振る舞い処理用共有ライブラリ名T2b、共有ライブラリの版数情報T2cは、図8と同様

なので説明は省略する。

【 0 0 6 3 】

フラグ T 2 d は、メモリ 4 2 にロード済みか否かを示す情報が記載され、ロード済みなら Y、そうでなければ N である。関数テーブルへのアドレス T 2 e は、版数情報に対応した分散オブジェクト振る舞い処理の関数テーブルのアドレスが記載される。

【 0 0 6 4 】

例えば、版数情報 ( 1 . 0 1 ) の分散オブジェクト振る舞い処理の関数テーブルのアドレスが 0x00012345 が示す関数テーブルは t ( 1 . 0 1 ) であり、版数情報 ( 1 . 0 2 ) の分散オブジェクト振る舞い処理の関数テーブルのアドレスが 0x00023456 が示す関数テーブルは t ( 1 . 0 2 ) である。

【 0 0 6 5 】

次に分散オブジェクト呼び出し制御手段 4 3 の動作をフローチャートを用いて説明する。図 1 1 は分散オブジェクト呼び出し制御手段 4 3 の動作手順を示すフローチャートである。

〔 S 1 0 〕 版数情報 A を含む電文を受信した場合、分散オブジェクト呼び出し制御手段 4 3 は、コンポーネント版数情報管理テーブル T 2 に対し、版数情報 A があるか否かを判断する。あればステップ S 1 1 へ、なければステップ S 1 2 へ行く。

〔 S 1 1 〕 分散オブジェクト呼び出し制御手段 4 3 は、版数情報 A の分散オブジェクトの関数を呼び出して実行する。

〔 S 1 2 〕 分散オブジェクト呼び出し制御手段 4 3 は、コンポーネント版数情報定義ファイル T 1 を参照して、版数情報 A があるか否かを判断する。あればステップ S 1 4 へなければステップ S 1 3 へ行く。

〔 S 1 3 〕 サーバ装置 4 0 は、クライアントへエラーを送信する。

〔 S 1 4 〕 分散オブジェクト呼び出し制御手段 4 3 は、分散オブジェクト格納リポジトリ 4 1 からメモリ 4 2 への動的リンクを行う。

〔 S 1 5 〕 分散オブジェクト呼び出し制御手段 4 3 は、コンポーネント版数情報管理テーブル T 2 の内容を変更する。

〔S 1 6〕分散オブジェクト呼び出し制御手段 4 3 は、版数情報 A の分散オブジェクトの関数を呼び出して実行する。

【0 0 6 6】

以上説明したように、本発明の分散処理システム 1 a は、サーバ装置内（サーバ・アプリケーション内）で分散オブジェクト呼び出し制御手段 4 3 を設けて、電文に含まれたシステム全体の版数情報に該当する分散オブジェクト振る舞い処理用共有ライブラリを動的にリンクする構成とした。さらに、分散オブジェクト振る舞い処理の関数名、変数名を C ++ 言語のネームスペースを使用する構成にした。

【0 0 6 7】

これにより、各版数の名前の衝突をなくし、実行中のトランザクションを停止することなく、同一プロセス上に複数の版数のソフトウェアを起動することができ。

【0 0 6 8】

また、クライアント・アプリケーションとサーバ・アプリケーションで、システム全体の版数情報を電文に含めて配信し合いながら、分散オブジェクトの置換を行う構成としたので、n 層のクライアント・サーバシステムに対しても複数の装置間で同期しながら分散オブジェクトの置換が可能になる。

【0 0 6 9】

次に第 2 の実施の形態の分散処理システムについて説明する。図 1 2 はコンポーネント版数情報管理テーブルの構成を示す図である。コンポーネント版数情報管理テーブル T 2 - 1 は、図 1 0 で説明したコンポーネント版数情報管理テーブル T 2 に、参照カウンタ T 2 f を設けたものである。

【0 0 7 0】

参照カウンタ T 2 f には、メモリ 4 2 に格納されている分散オブジェクト振る舞い処理の参照回数をカウントした値が記載される。

分散オブジェクト呼び出し制御手段 4 3 は、分散オブジェクト振る舞い処理の関数を呼び出す時に、参照カウンタ T 2 f に 1 だけ加算し、その関数が復帰した時に、参照カウンタ T 2 f から 1 だけ減算するようにする。



## 【 0 0 7 1 】

図 1 3 は分散オブジェクトの解放処理手順を示すフローチャートである。

〔 S 2 0 〕 分散オブジェクト呼び出し制御手段 4 3 は、分散オブジェクト振る舞い処理を実行して完了すると、参照カウンタ T 2 f から 1 だけ減算する。

〔 S 2 1 〕 分散オブジェクト呼び出し制御手段 4 3 は、参照カウンタ T 2 f の値が 0 か否かを判断する。0 ならステップ S 2 2 へ、0 でなければステップ S 2 4 へ行く。

〔 S 2 2 〕 参照カウンタ T 2 f の値が 0 である版数情報より、新しい版数情報がコンポーネント版数情報管理テーブル T 2 内にあるか否かを判断する。新しい版数情報がなければステップ S 2 3 へ、新しい版数情報があればステップ S 2 4 へ行く。〔 S 2 3 〕 メモリ 4 2 から参照カウンタ T 2 f の値が 0 である分散オブジェクト振る舞い処理を解放（削除）する。

〔 S 2 4 〕 実行中の処理を継続する。

## 【 0 0 7 2 】

以上説明したように、第 2 の実施の形態では、分散オブジェクト振る舞い処理に対する参照カウンタが 0 であり、かつ、その版数情報より新しい版数情報がコンポーネント版数情報管理テーブル T 2 にある場合には、その分散オブジェクト振る舞い処理をメモリ 4 2 から解放する構成にした。

## 【 0 0 7 3 】

これにより、旧版の分散オブジェクト振る舞い処理の利用者がなくなった時には、自動的にメモリ 4 2 から解放されることになり、分散オブジェクト振る舞い処理の効率のよい置換が可能になる。

## 【 0 0 7 4 】

次に第 3 の実施の形態の分散処理システムについて説明する。図 1 4 は電文の構成例を示す図である。電文 2 - 1 は、分散オブジェクトの関数名 2 a と、システム全体の版数を示す版数情報 2 b と、パラメタ 2 c と、評価中フラグ 2 d から構成される。この評価中フラグ 2 d は、分散オブジェクト振る舞い処理が評価中か否かを示すものである。

## 【 0 0 7 5 】

図15は電文のIDL定義例を示す図であり、図16はIDL定義からC++言語へのマッピング例を示す図である。図に示すように、第3の実施の形態では版数情報の他に、評価中フラグを含む。

#### 【0076】

図17は評価対象の分散オブジェクトの処理手順を示すフローチャートである。

〔S30〕分散オブジェクト呼び出し制御手段43は、版数情報Aの電文中の評価中フラグが評価中か否かを判断する。評価中であればステップS31へ、評価中でなければ図11のステップS10へ行く。

〔S31〕分散オブジェクト呼び出し制御手段43は、コンポーネント版数情報定義ファイルT1に対し、版数情報Aがあるか否かを判断する。あればステップS33へ、なければステップS32へ行く。

〔S32〕サーバ装置40は、クライアントへエラーを送信する。

〔S33〕分散オブジェクト呼び出し制御手段43は、該当する分散オブジェクト振る舞い処理を動的にリンクするのではなく、動的に別のプロセスを生成し、その別プロセス内のスケルトン組み込み機構を実行する。そして、該当するスケルトン処理用共有ライブラリと分散オブジェクト振る舞い処理を別プロセスへロードする。

〔S33〕分散オブジェクト呼び出し制御手段43は、コンポーネント版数情報管理テーブルT2の内容を変更する。

〔S35〕別プロセス内で、評価中の分散オブジェクト振る舞い処理を実行する。

#### 【0077】

以上説明したように、第3の実施の形態では、評価中の分散オブジェクト振る舞い処理は別プロセスで実行する構成とした。これにより、評価中の分散オブジェクト振る舞い処理が他の分散オブジェクト振る舞い処理に与える影響をなくし、分散オブジェクトの置換制御の品質を向上することが可能になる。

#### 【0078】

次に第4の実施の形態の分散処理システムについて説明する。図18はコンポ

ーネント版数情報管理テーブルの構成を示す図である。コンポーネント版数情報管理テーブル T 2 - 2 は、図 1 2 で説明したコンポーネント版数情報管理テーブル T 2 - 1 に、評価カウンタ T 2 g を設けたものである。

評価カウンタ T 2 g は、別プロセスのメモリに格納されている分散オブジェクト振る舞い処理の評価回数をカウントした値が記載される。

【 0 0 7 9 】

分散オブジェクト呼び出し制御手段 4 3 では、別プロセスを生成して、評価中の分散オブジェクト振る舞い処理をリンクした場合に評価カウンタを 1 だけ加算し、評価処理の完了応答を受信した時に、評価カウンタから 1 だけ減算するようにする。

【 0 0 8 0 】

図 1 9 は分散オブジェクトの評価処理手順を示すフローチャートである。

〔 S 4 0 〕 分散オブジェクト呼び出し制御手段 4 3 は、評価中の分散オブジェクト振る舞い処理を実行して、応答を受信すると、評価カウンタ T 2 g から 1 だけ減算する。

〔 S 4 1 〕 分散オブジェクト呼び出し制御手段 4 3 は、評価カウンタ T 2 g の値が 0 か否かを判断する。0 ならステップ S 4 2 へ、0 でなければステップ S 4 4 へ行く。

〔 S 4 2 〕 フラグ T 2 d が Y なら（動的にリンクしているなら）ステップ S 4 3 へ、N なら（動的にリンクしていないなら）ステップ S 4 4 へ行く。

〔 S 4 3 〕 別プロセスを停止する。

〔 S 4 4 〕 実行中の処理を継続する。

【 0 0 8 1 】

以上説明したように、第 4 の実施の形態では、評価中フラグが 0 である場合には、評価中の分散オブジェクト振る舞い処理を実行していた別プロセスを停止する構成にした。これにより、評価用のプロセスは、利用者がなくなった時に、自動的に停止することが可能になる。

【 0 0 8 2 】

次に本発明のプログラム置換方法について説明する。図 2 0 は本発明のプロゲ

ラム置換方法の処理手順を示すフローチャートである。

〔S 5 0〕 クライアントからプログラムの版数情報を含む電文を送信する。

〔S 5 1〕 電文をサーバで受信する。

〔S 5 2〕 置換用のプログラムを格納する。

〔S 5 3〕 プログラムの情報を転送前情報管理テーブルで管理する。

〔S 5 4〕 電文に含まれる版数情報に対応するプログラムを、起動中のプログラムが存在するプロセス内のメモリに動的にリンクして、プログラムを呼び出して実行する。

〔S 5 5〕 メモリに格納されているプログラムの情報を転送後情報管理テーブルで管理する。

〔S 5 6〕 他装置内にあるプログラムを呼び出して実行するために、版数情報を含む電文の配信処理を行うことにより、n 階層間でのプログラムの置換を、装置間で同期させて行う。

#### 【 0 0 8 3 】

転送後情報管理テーブルは、メモリに格納されているプログラムの参照回数をカウントする参照カウンタをテーブル項目として有する。そして、プログラムに対する参照カウンタの値が 0 で、プログラムの版数情報より新しい版数情報が転送後情報管理テーブルにある場合には、プログラムをメモリから削除する。

#### 【 0 0 8 4 】

さらに、プログラムの版数情報と、プログラムが評価中か否かを示す評価フラグと、を含む電文を送信する。そして、評価フラグが評価中を示す場合には、別プロセスの中で評価対象のプログラムを呼び出して実行する。

#### 【 0 0 8 5 】

また、転送後情報管理テーブルは、メモリに格納されているプログラムの評価回数をカウントする評価カウンタをテーブル項目として有する。そして、プログラムに対する評価カウンタの値が 0 の場合は、別プロセスの中のプログラムの実行を中止する。

#### 【 0 0 8 6 】

さらにまた、管理サーバにより、置換用のプログラム及びプログラムに関する

情報の設定・管理を一括して行う。

以上説明したように、本発明のプログラム置換システム、分散処理システム及びプログラム置換方法によれば、分散オブジェクト・コンピューティングを基本としたn階層クライアント／サーバ環境に対し、トランザクション処理を全く中断することなく、新しいソフトウェアを動的に組み込むことができる。

【0087】

また、新・旧のソフトウェアを混在させながら、複数のコンピュータに分散した複数のソフトウェアを段階的に入れ替えることが可能となる。

さらに、古い版数のソフトウェアを使用するクライアントのトランザクション処理の完了を確実に待ち、不要となるメモリを解放することで、システム資源を有効活用することができる。

【0088】

また、ソフトウェアが評価中の場合には、すでにトランザクション処理を実行中のプロセスで新しいソフトウェアを実行するのではなく、別プロセスにて分散オブジェクトの振る舞い処理を実行するため、信頼性を低下させずに、新しいソフトウェアを導入することができる。

【0089】

さらに、評価が完了したソフトウェアは、自動的にプロセス内にロードされ、トランザクションが開始される。また、別プロセスで実行されたトランザクション処理の完了を待ち、不要となるプロセスを消滅させることで、システム資源を有効に活用することができる。

【0090】

さらに、ソフトウェアをリポジトリサーバに集中管理し、ソフトウェア転送することで、n階層クライアント／サーバ環境で煩雑となるソフトウェア更新作業の簡略化、自動化、及び作業品質の向上を実現することが可能になる。

【0091】

なお、上記の説明では、ORBとしてCORBAを用いたが、これ以外のORBのネットワークを使用してもよい。また、上記の説明では、n階層のクライアント／サーバ・システムを中心に説明したが、多様な分野の情報通信ネットワー

クのソフトウェアの動的置換に対して、本発明を適用可能である。

【0092】

【発明の効果】

以上説明したように、本発明のプログラム置換システムは、プログラムの版数情報を含む電文を送信し、この版数情報に対応するプログラムをプロセス上に動的にリンクさせて実行する構成とした。これにより、現在行っている業務のソフトウェアを実行中に、新しい版数のソフトウェアを組み込むことができるので、業務の中断なしにソフトウェアを動的に効率よく置換することが可能になる。

【0093】

また、本発明の分散処理システムは、分散オブジェクトの版数情報を含む電文を送信し、この版数情報に対応する分散オブジェクトをプロセス上に動的にリンクさせて実行する構成とした。これにより、現在行っている業務のソフトウェアを実行中に、新しい版数のソフトウェアを組み込むことができるので、業務の中断なしにソフトウェアを動的に効率よく置換することが可能になる。

【0094】

さらに、本発明のプログラム置換方法は、プログラムの版数情報を含む電文を送信し、この版数情報に対応するプログラムをプロセス上に動的にリンクさせて実行することとした。これにより、現在行っている業務のソフトウェアを実行中に、新しい版数のソフトウェアを組み込むことができるので、業務の中断なしにソフトウェアを動的に効率よく置換することが可能になる。

【図面の簡単な説明】

【図1】

本発明のプログラム置換システムの原理図である。

【図2】

プログラム置換システムの動作手順を示すフローチャートである。

【図3】

分散処理システムの構成を示す図である。

【図4】

サーバ装置の構成を示す図である。

【図 5】

電文の構成例を示す図である。

【図 6】

電文の I D L 定義例を示す図である。

【図 7】

I D L 定義から C + + 言語へのマッピング例を示す図である。(A) は版数情報 (1. 0 1) であり、(B) は版数情報 (1. 0 2) である。

【図 8】

コンポーネント版数情報定義ファイルの構成を示す図である。

【図 9】

ディレクトリの構成例を示す図である。

【図 1 0】

コンポーネント版数情報管理テーブルの構成を示す図である。

【図 1 1】

分散オブジェクト呼び出し制御手段の動作手順を示すフローチャートである。

【図 1 2】

コンポーネント版数情報管理テーブルの構成を示す図である。

【図 1 3】

分散オブジェクトの解放処理手順を示すフローチャートである。

【図 1 4】

電文の構成例を示す図である。

【図 1 5】

電文の I D L 定義例を示す図である。

【図 1 6】

I D L 定義から C + + 言語へのマッピング例を示す図である。

【図 1 7】

評価対象の分散オブジェクトの処理手順を示すフローチャートである。

【図 1 8】

コンポーネント版数情報管理テーブルの構成を示す図である。

【図 1 9】

分散オブジェクトの評価処理手順を示すフローチャートである。

【図 2 0】

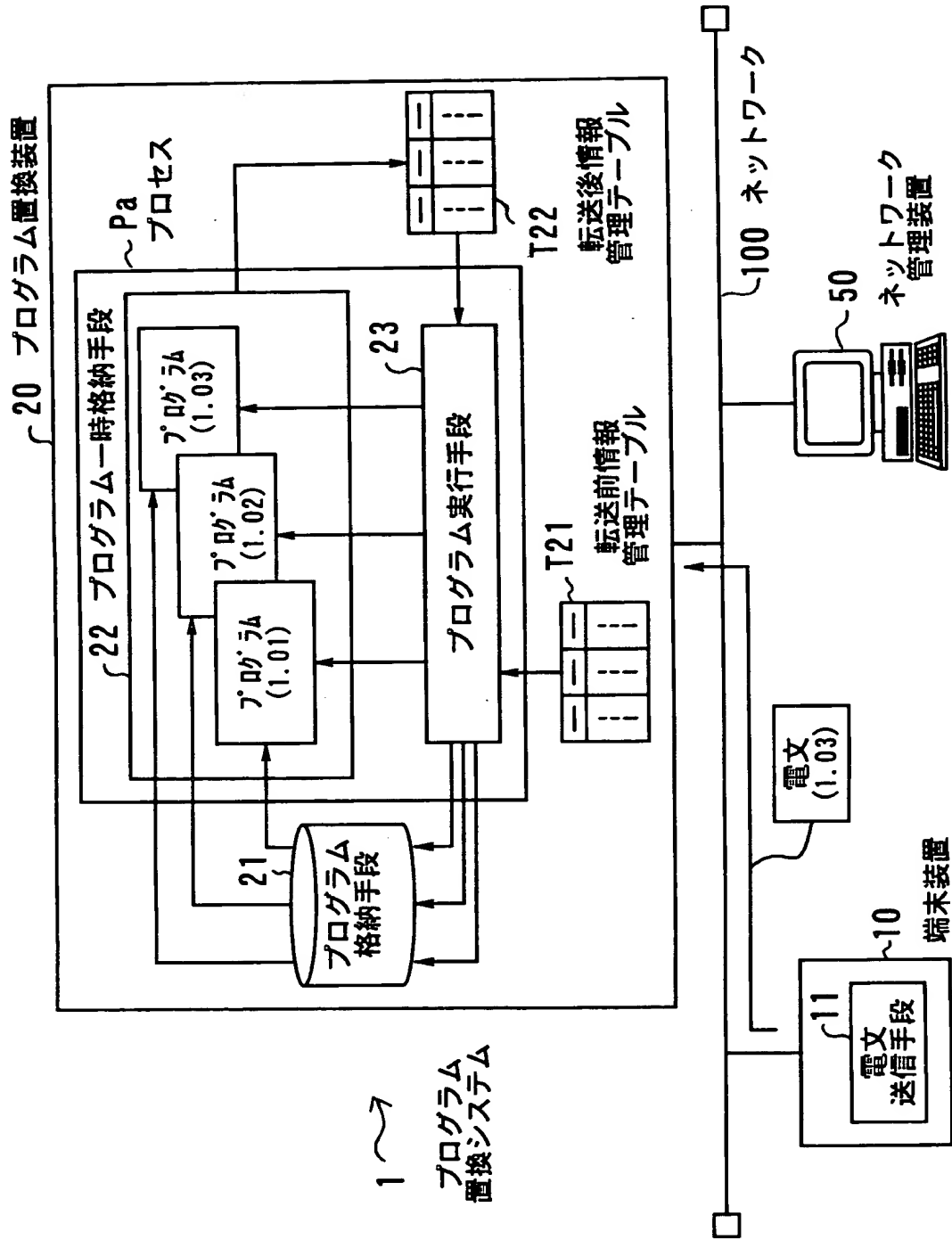
本発明のプログラム置換方法の処理手順を示すフローチャートである。

【符号の説明】

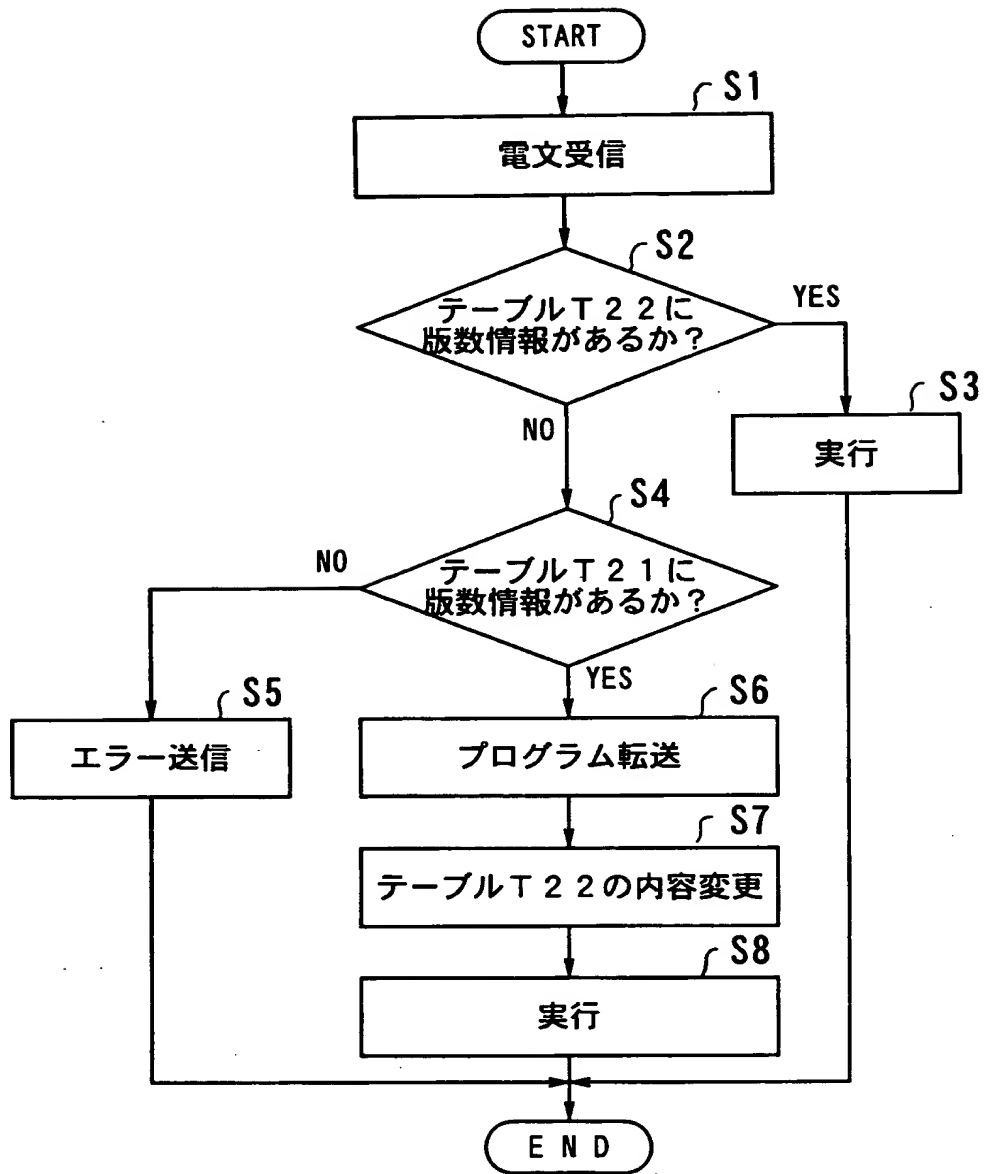
- 1 プログラム置換システム
- 1 0 端末装置
- 1 1 電文送信手段
- 2 0 プログラム置換装置
- 2 1 プログラム格納手段
- 2 2 プログラム一時格納手段
- 2 3 プログラム呼び出し制御手段
- T 2 1 転送前情報管理テーブル
- T 2 2 転送後情報管理テーブル
- Pa プロセス



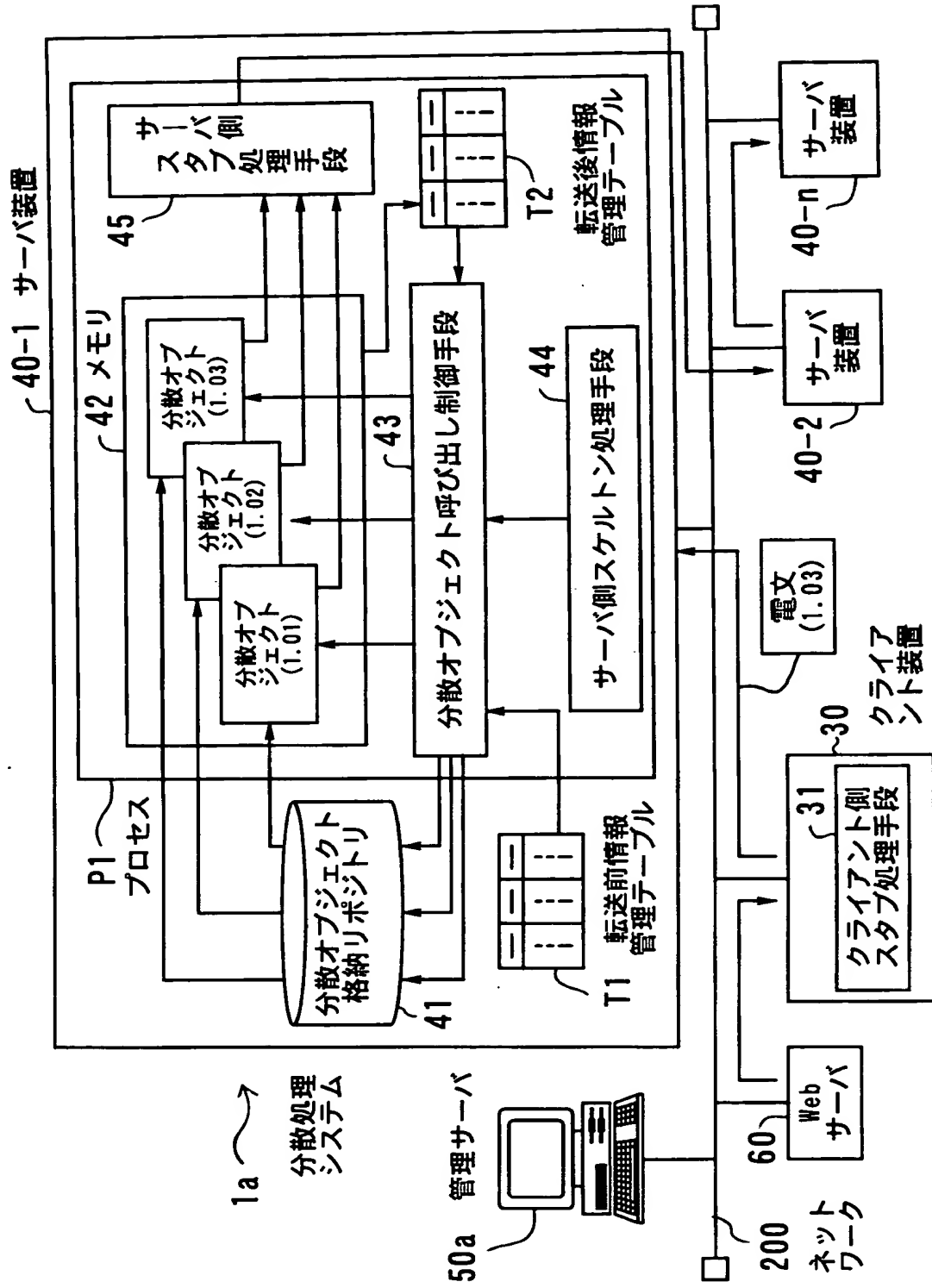
【書類名】 図面  
【図 1】



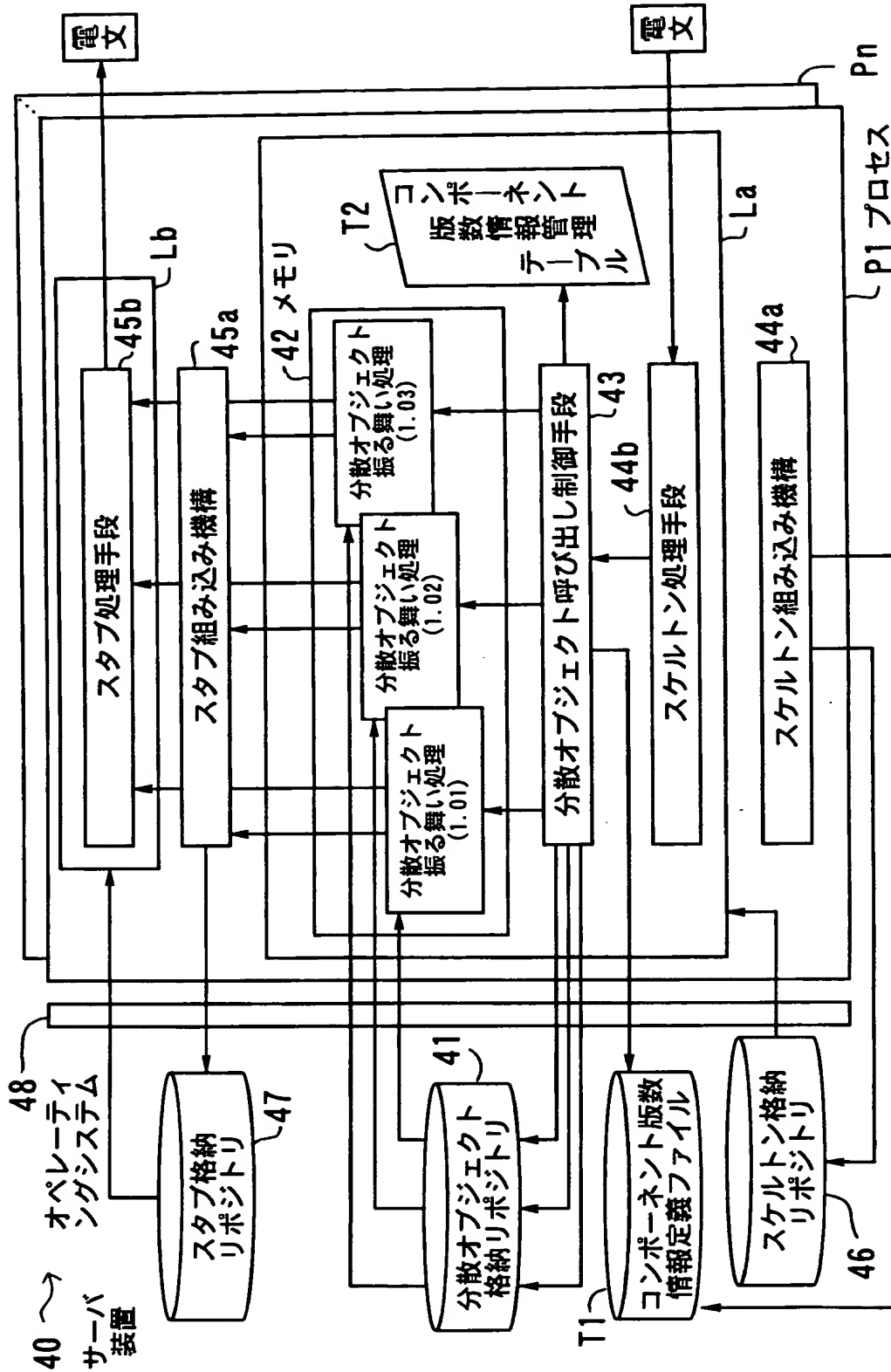
【図 2】



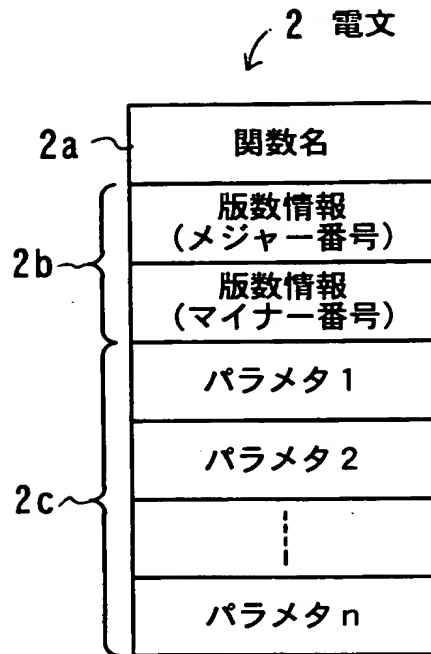
【図 3】



【図 4】



【図 5】



【図 6】

```

module XXX {
  interface YYY {
    void foo (in unsigned long major_version,
              in unsigned long minor_version,
              ...parameter_1
              ...
              ...parameter_n);
    ...
  };
};

```

版数情報

【図 7】

(A)

```

Na → namespace XXX 1_01 {
    class XXX {
        class YYY {
            public:
                void foo (CORBA::ULong major_version,
                           CORBA::ULong minor_version,
                           ...parameter_1
                           ...
                           ...parameter_n);
            ...
        };
    };
};

```

(B)

```

Nb → namespace XXX 1_02 {
    class XXX {
        class YYY {
            public:
                void foo (CORBA::ULong major_version,
                           CORBA::ULong minor_version,
                           ...parameter_1
                           ...
                           ...parameter_n);
            ...
        };
    };
};

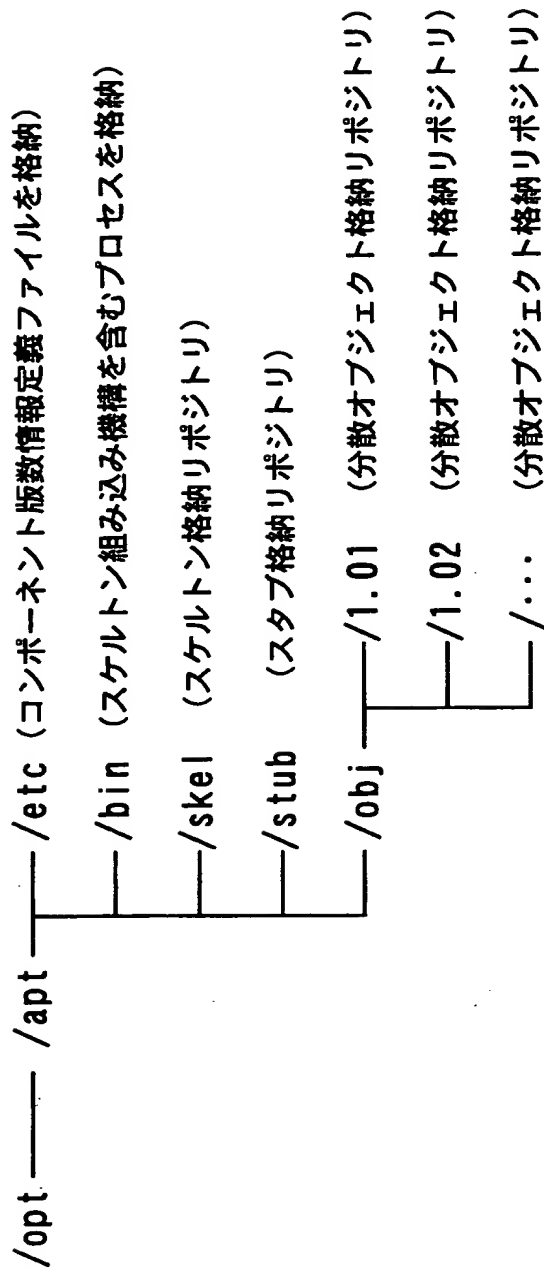
```

【図 8】

↙ T1 コンポーネント版数情報定義ファイル

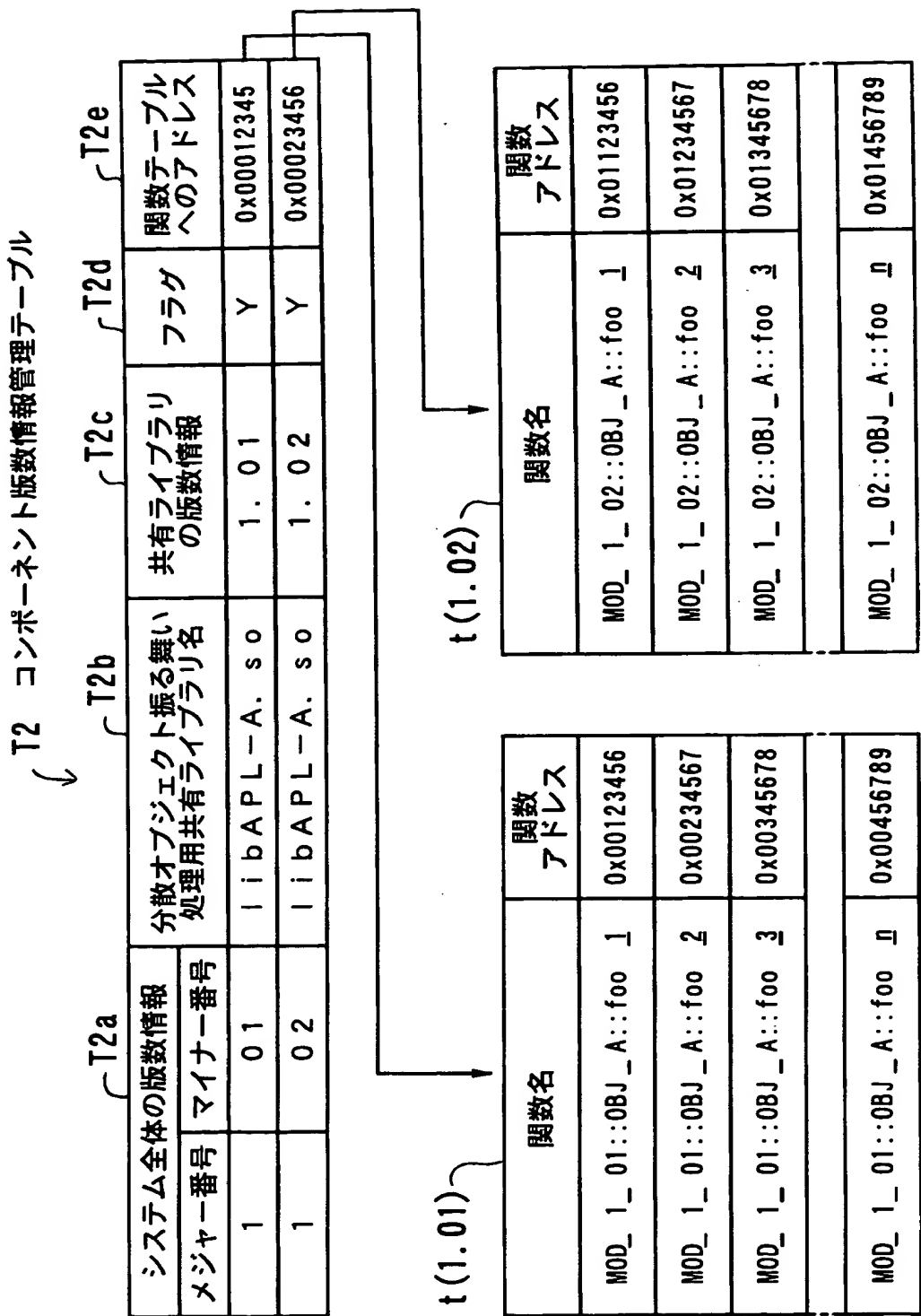
T1a		T1b		T1c	T1d
システム全体の版数情報		分散オブジェクト振る舞い 処理用共有ライブラリ名		共有ライブラリ の版数情報	プロセス名
メジャー番号	マイナー番号				
1	01	libAPL-A. so		1. 01	process-1
1	01	libAPL-B. so		1. 01	process-2
1	01	libAPL-C. so		1. 01	process-2
1	01	libAPL-X. so		1. 01	process-n
1	02	libAPL-A. so		1. 02	process-1
1	02	libAPL-B. so		1. 01	process-2
1	02	libAPL-C. so		1. 02	process-2
1	02	libAPL-X. so		1. 02	process-n
1	03	libAPL-A. so		1. 02	process-1
1	03	libAPL-B. so		1. 02	process-2
1	03	libAPL-C. so		1. 03	process-2
1	03	libAPL-X. so		1. 03	process-n

【図 9】

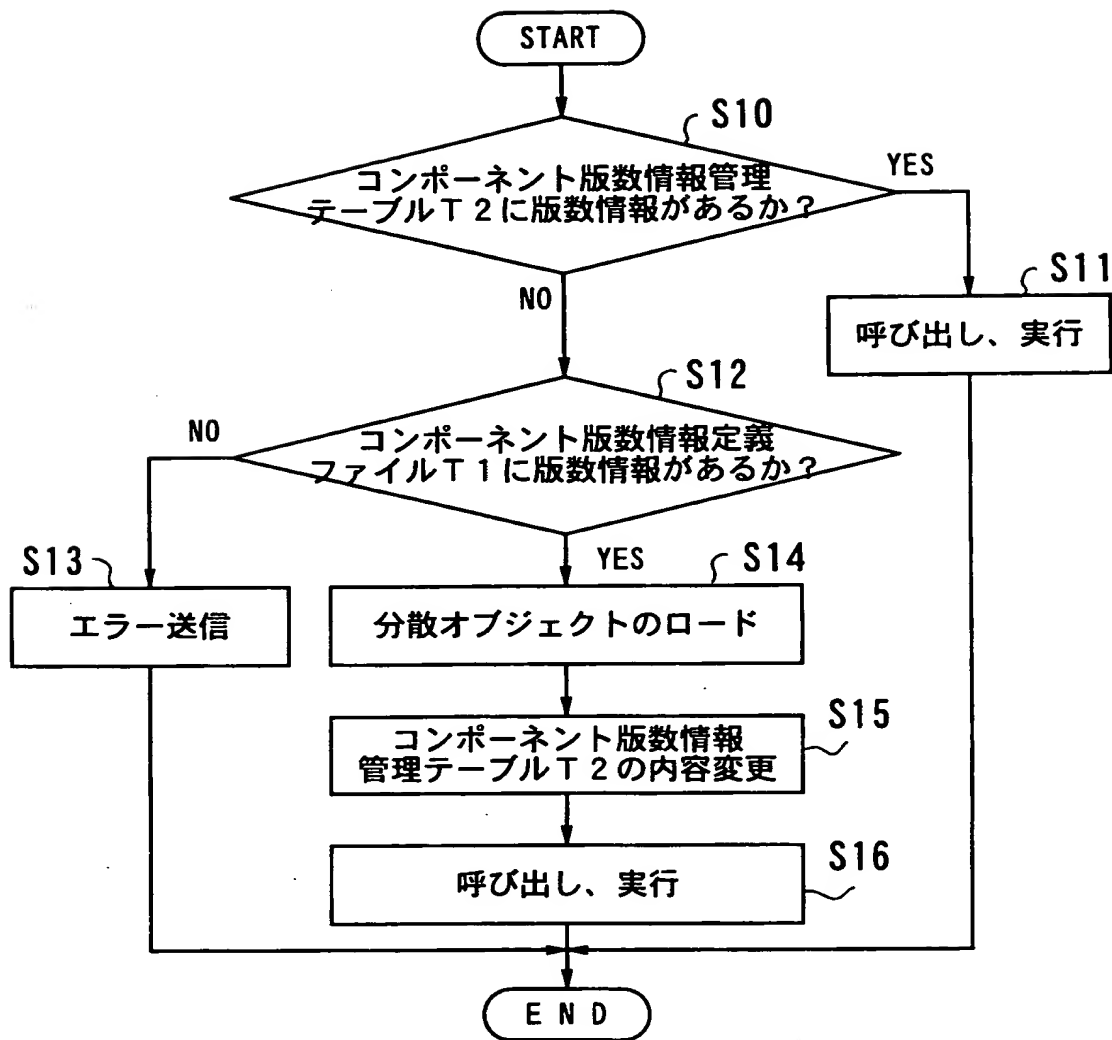




【図10】



【図 11】



【図 1 2】

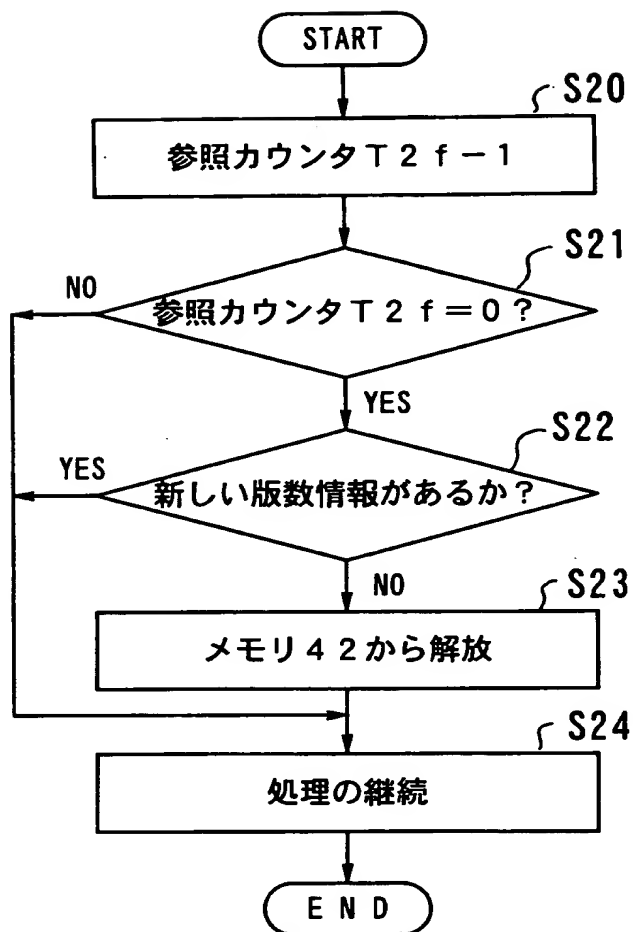
コンポーネント版数情報管理テーブル

T2a		T2b		T2c	T2d	T2f	T2e
システム全体の版数情報		分散オブジェクト振る舞い 処理用共有ライブラリ名		共有ライブラリ の版数情報	フラグ	参照カ ウンタ	関数テーブル へのアドレス
メジャー番号	マイナー番号						
1	01	libA PL-A. so		1. 01	N	0	0x0
1	02	libA PL-A. so		1. 02	Y	12	0x00012345
1	03	libA PL-A. so		1. 03	Y	7	0x00023456

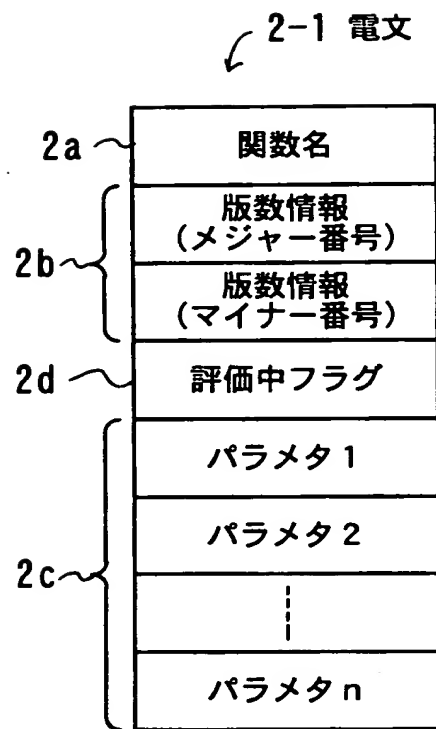
  

t(1.02)		t(1.03)	
関数名	関数 アドレス	関数名	関数 アドレス
MOD_1_02::OBJ_A::foo 1	0x00123456	MOD_1_03::OBJ_A::foo 1	0x01123456
MOD_1_02::OBJ_A::foo 2	0x00234567	MOD_1_03::OBJ_A::foo 2	0x01234567
MOD_1_02::OBJ_A::foo 3	0x00345678	MOD_1_03::OBJ_A::foo 3	0x01345678
MOD_1_02::OBJ_A::foo 4	0x00456789	MOD_1_03::OBJ_A::foo 4	0x01456789

【図 1 3】



【図 1 4】



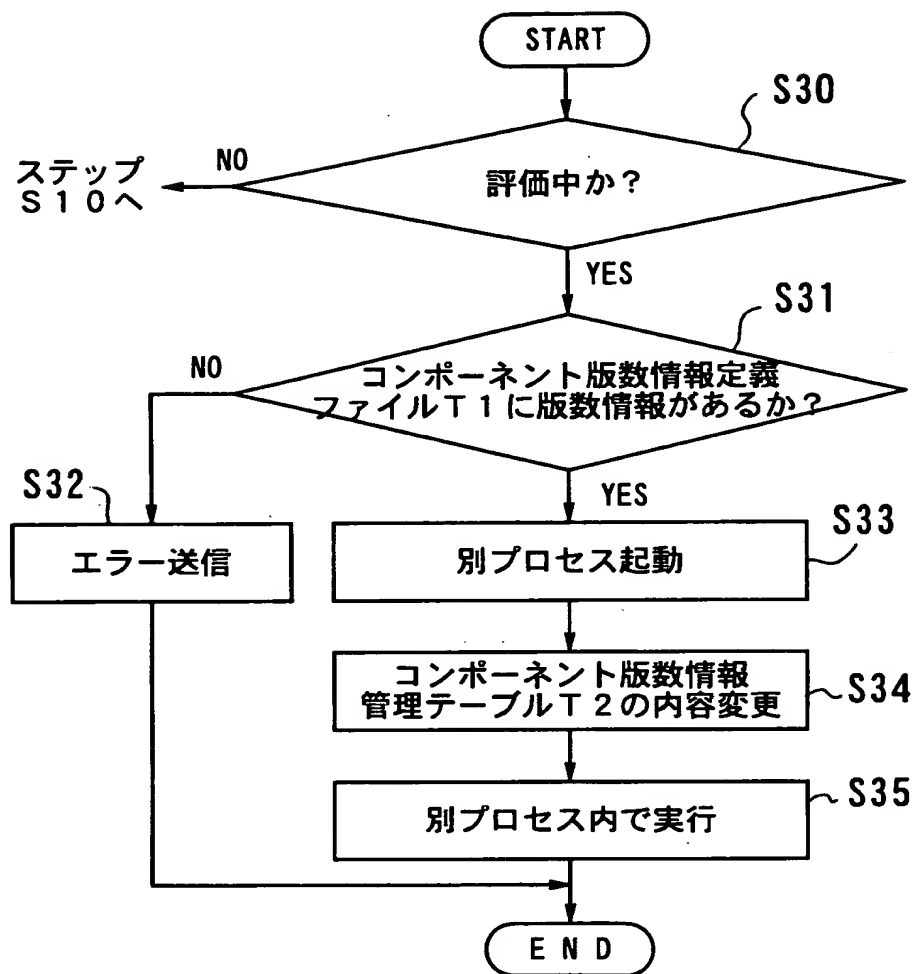
【図 1 5】

```
module XXX {  
    interface YYY {  
        void foo (in unsigned long major_version,  
                  in unsigned long minor_version,  
                  in boolean eval, ← 評価中フラグ  
                  ...parameter _1  
                  ...  
                  ...parameter _n);  
        ...  
    };  
};
```

【図 16】

```
namespace XXX 1_01 {  
  class XXX {  
    class YYY {  
    public:  
      void foo (CORBA::ULong major_version,  
                CORBA::ULong minor_version,  
                CORBA::Boolean eval, ←  
                ...parameter_1      評価中フラグ  
                ...  
                ...parameter_n);  
      ...  
    };  
  };  
};
```

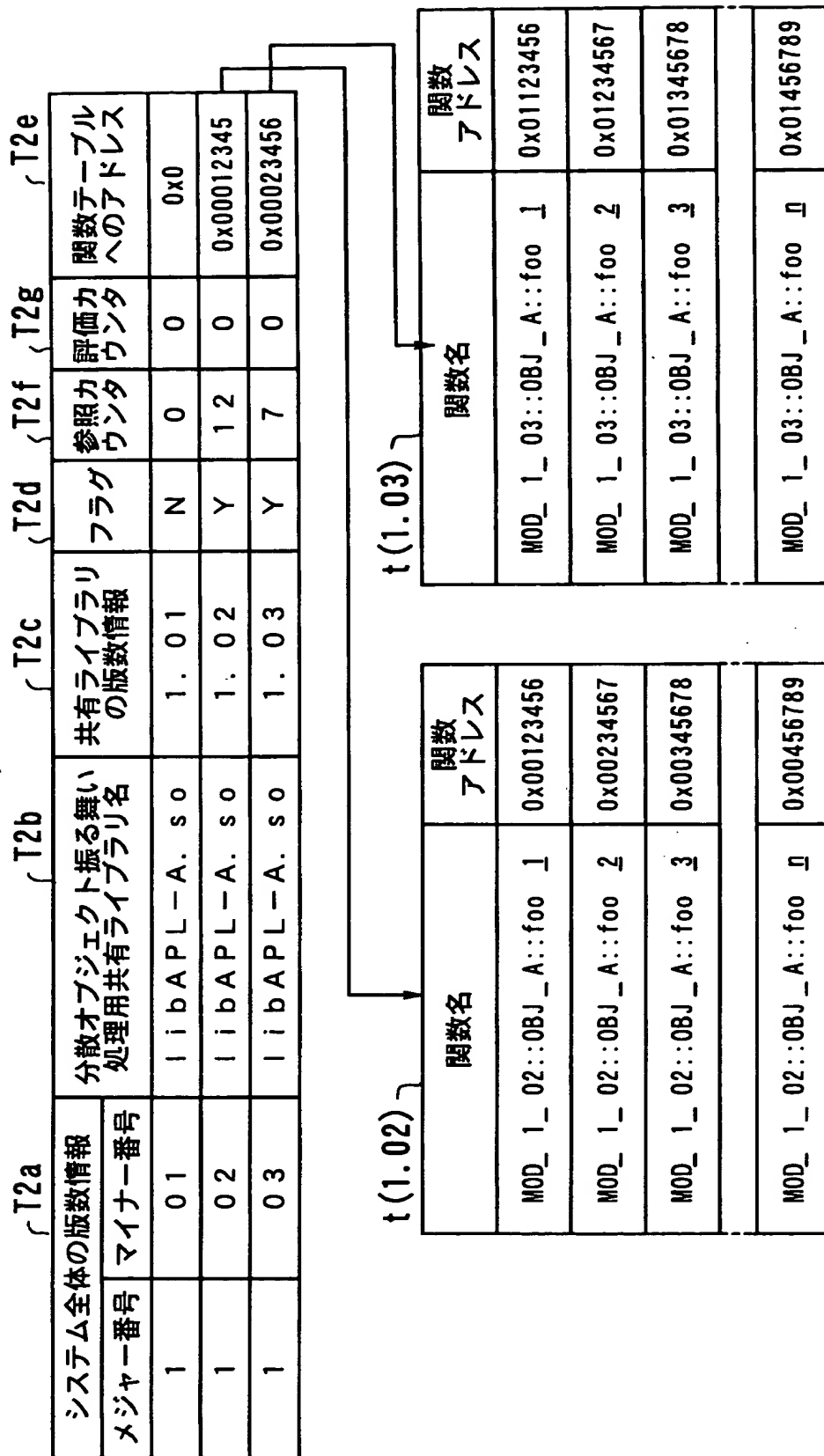
【図 17】



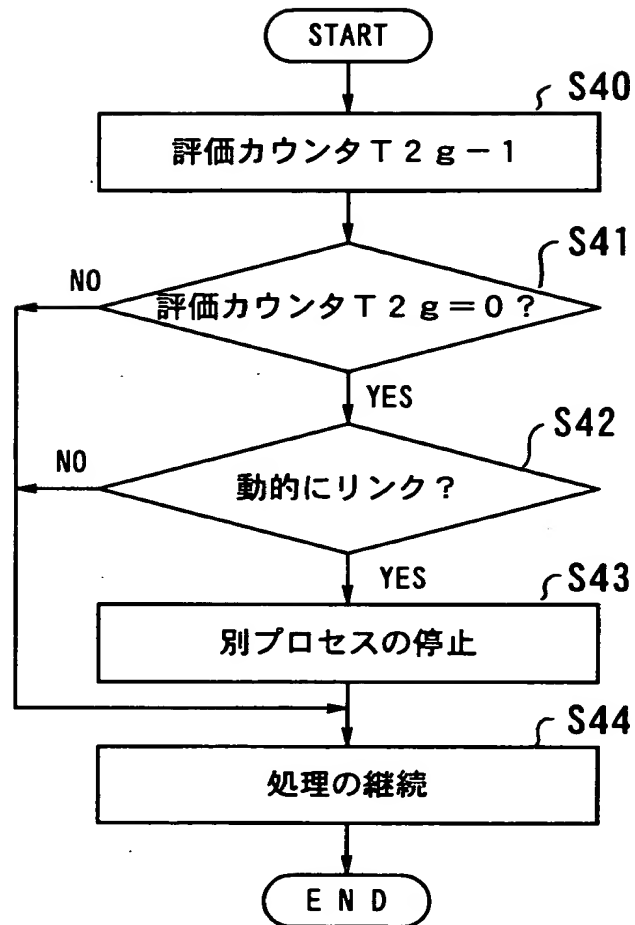


【図 1 8】

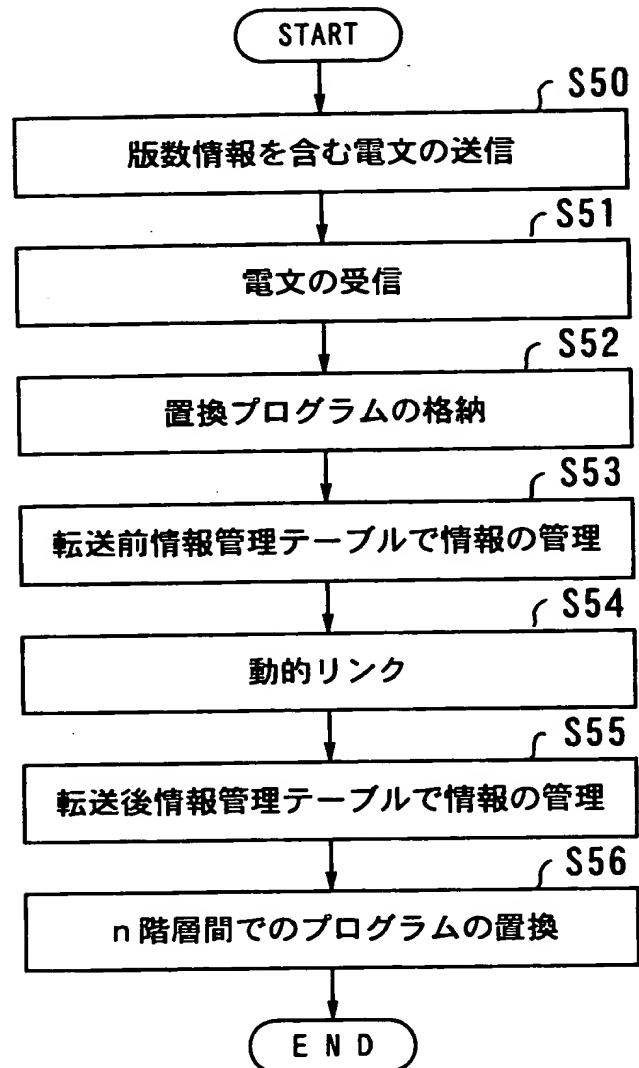
T2-2 コンポーネント版数情報管理テーブル



【図 19】



【図 2 0】



【書類名】            要約書

【要約】

【課題】    業務の中断なしにソフトウェアを動的に効率よく置換する。

【解決手段】    電文送信手段 1 1 は、プログラムの版数情報を含む電文を送信する。プログラム格納手段 2 1 は、置換用のプログラムを格納する。転送前情報管理テーブル T 1 は、プログラムの情報を管理する。プログラム一時格納手段 2 2 は、プログラム格納手段 2 1 から転送されたプログラムを一時格納する。転送後情報管理テーブル T 2 2 は、プログラム一時格納手段 2 2 に格納されているプログラムの情報を管理する。プログラム実行手段 2 3 は、電文に含まれる版数情報に対応するプログラムを、起動中のプログラムが存在するプロセス P a 内のプログラム一時格納手段 2 2 に動的にリンクし、プログラムを実行する。

【選択図】            図 1

出 願 人 履 歴 情 報

識別番号 [000005223]

1. 変更年月日 1996年 3月26日  
[変更理由] 住所変更  
住 所 神奈川県川崎市中原区上小田中4丁目1番1号  
氏 名 富士通株式会社